Dynamically Reconfigurable Neuron Architecture for the Implementation of Self-Organizing Learning Array

Janusz A. Starzyk and Yongtao Guo
School of Electrical Engineering & Computer Science
Ohio University, Athens, OH 45701
{starzyk, gyt}@bobcat.ent.ohiou.edu

Abstract: In this paper, we describe a new dynamically reconfigurable neuron hardware architecture based on modified Xilinx Picoblaze microcontroller and self-organizing learning array (SOLAR) algorithm reported earlier. This architecture is aiming at hundreds of traditional reconfigurable field programmable gate arrays (FPGAs) used to build SOLAR learning machine that has many advantages over traditional neural network hardware implementation. Neurons are optimized for area and speed, and the whole system is dynamically self-reconfigurable during the runtime. The system architecture is expandable to a large multiple-chip system.

Keywords:
Dynamical Reconfiguration, Self-Organization, Picoblaze, FPGA.

1. Introduction

The immense computing power of the brain is believed to be the result of the parallel and distributed computing performed by approximately 10^{11} neurons, each with an average of $10^3 – 10^4$ connections. To prototype networks that mimic the biological neurons using hardware, currently we can benefit from the flexibility of FPGAs used as general-purpose processors. During the last decade, numbers of approaches have been attempted for the application of reconfigurable hardware to the neural networks [1,2]. However, these implementations are focused on interconnect weight dominated architectures which are not easily expandable to multiple FPGAs with identical cell architecture. Additionally, often their connections and functionalities are fixed and predefined by off-line simulation. One exception here are FPGA based arrays of identical cells to implement evolutionary computing and genetic algorithms [3,4,5]. Also, recently cellular neural networks used in image processing [6] and their hardware implementation have attracted a lot of attention.

In this paper, we present a dynamically reconfigurable (i.e. during runtime), data-based and expandable neuron architecture to implement a self-organizing learning array (SOLAR). The neuron's functionality is represented in the form of flexible connections and organization of an array of evolvable signal processing blocks. Each neuron can implement various arithmetic and logic functions. The neurons can self-reconfigure and use local interconnect for maximum performance.

The rest of the paper is organized as follows. Section 2 talks about our pervious work on SOLAR. Section 3 deals with the architecture of the dynamically reconfigurable neuron. Section 4 reports the simulation and experimental results. Finally, Section 5 concludes this paper.

2. Previous Work

Self-organization is important in artificial neural networks (ANNs) and machine learning. Previously, a self-organizing learning algorithm that combines neural networks and information theory was presented in [7]. This entropy-based neural network learning algorithm was simulated on standard benchmarks and proved to be advantageous over many existing neural networks and machine learning algorithms in a wide range of technical applications, in particular, dealing with noisy or incomplete data. Based on this algorithm, we proposed SOLAR system which is different from classical ANNs in the way it is organized and how it learns. SOLAR self-organizes its hardware resources to perform classification and recognition tasks. Similar to the structure of cellular neural networks, SOLAR has a fixed array of elemental processing units acting as single neurons, and programmable interconnections between them. Initially, SOLAR neurons are randomly connected to previously generated neurons. They learn adaptively using both primary inputs and inputs from other neurons. Controlled by signals from other neurons, they perform basic transformations of their input signals. A neuron

parameters and connections are re-configured as a result of training, and effectively, SOLAR's structure self-organizes establishing its final wiring.

Earlier work mainly focused on SOLAR algorithm simulation [7] and its hardware implementation was limited to a single FPGA chip with the cooperation from software [8]. This hardware implementation explored the adaptability of SOLAR to FPGA implementation, although the cellular neuron architecture was not fully explored since we utilized a shared block memory for all neurons. In SOLAR simulation, we adopt feed forward network structure for its stability and fast learning. SOLAR architecuture is divided into three main layers as shown in Fig. 1– input, processing and output layers. The interconnections are randomly initialized at the beginning of training. The basic building blocks of the architecture are the small neurons, trained using the entropy-based algorithm [7].

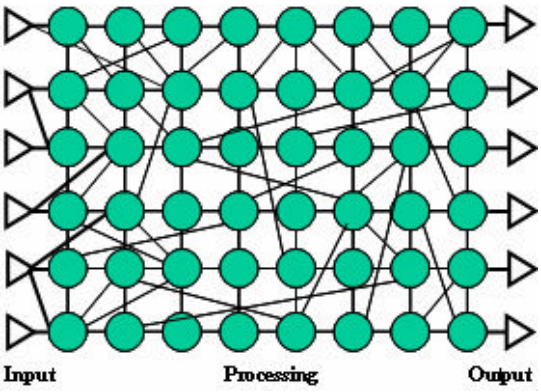

Fig. 1 Basic SOLAR structure

3. Neuron Architecture

The hardware architecture presented in this paper is based on identical neuron modules. The constant Coded Programmable State Machine (KCPSM)[9], an 8-bit micro-controller developed by Xilinx Corp., has been modified and embedded into the neuron module. The module contains circuits to be reprogrammed dynamically and to execute new programs without affecting other neurons' executions. A block level of a single neuron architecture is shown on Fig.2.

The dynamical programming ability is implemented by a dual-port 256x16-bit memory. The KCPSM reads the current program on one port while the other port can be used to store the new program. The two ports of the dual-port RAM operate independently,

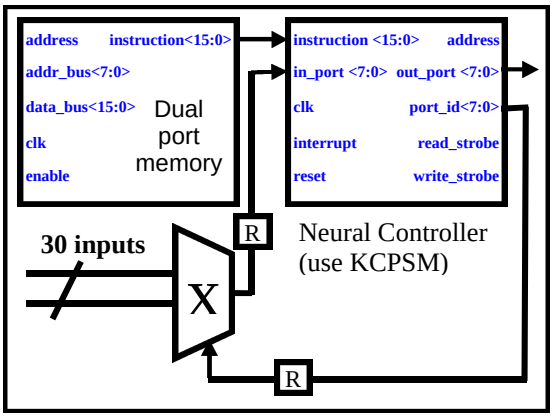

Fig.2 Single neuron's schematic

and the operation is via shared programming bus among all neurons. Therefore, the self-reconfiguration process can be performed affecting only the current neuron. The rest of the neurons inside the chip work with no interruption. The configuration time and contents can be controlled by software outside the chip or configuration data that is located in distributed memory cells in the system. The neuron inputs are from either primary inputs or any other neurons via a 30 to 1 multiplexer. The selection signals are decided by the content of the programming dual-port memory via execution of the programming commands for the particular neuron.

The single neuron architecture is expanded to multiple neurons in a single FPGA chip. In this work, we use a Nallatech board with Xilinx Virtex XCV800 FPGA [10]. It can contain an array of up to 28 neurons organized as shown in Fig.3 (The 480 Virtex XCV1000 FPGAs we are using to build 3D learning machine can contain up to 32 neurons on each chip.)

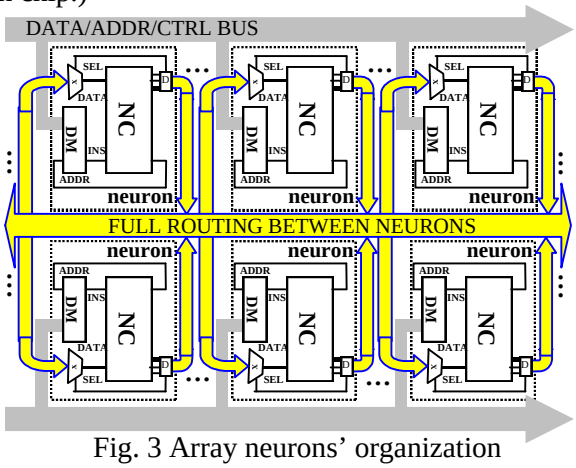

Fig. 3 Array neurons' organization

These neurons are fully connected via the connection bus. The neurons connections are decided by the

programming contents of each neuron. The programming contents can be dynamically updated via the configuration bus or set locally by a neuron. The configuration bus used to configure every single neuron is divided into 16-bit data, 8-bit address and 5-bit neuron selection buses. To demonstrate the functionality of the 28 neurons, we integrate a PCI interface controllers to transfer the data/configuration via the PCI bus to neurons.

4. Simulation and Results

In this section, results from prototyping a simple SOLAR architecture onto a single VIRTEX FPGA chip are discussed. The neurons self-organizing learning process can be configured during chip programming or dynamically updated while running. To demonstrate this process, a simple example is given to illustrate the implementation step of SOLAR algorithm. The implementation of the whole SOLAR array is organized in a similar way. In this simulation example, we have six out of twenty-eight neurons configured as shown in Fig.4.

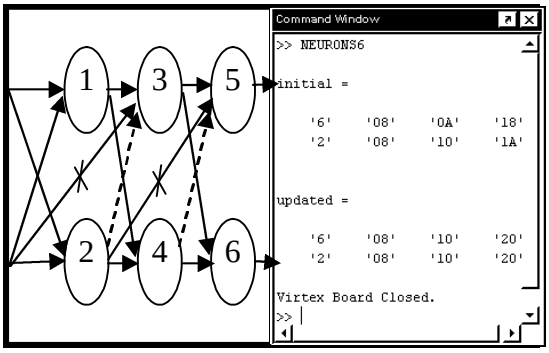

Fig.4 Experiment and result

The initial connections are shown in solid lines. Every neuron simply adds two inputs together, for instance, neuron 1 adds input 1 and 2 to its content; neuron 3 adds input 2 and the output of neuron 1; neuron 5 adds the outputs of neuron 2 and 3, etc. Later, we dynamically reconfigure the connections of neuron 3 and neuron 5. So neuron 3 has inputs from the outputs of neuron 1 and 2, and neuron 5 has inputs from the outputs of neuron 3 and 4 shown by the dotted lines. The results read out from the chip via PCI bus are shown in the Matlab console. In the inserted Matlab command console in Fig.4, "initial" values show primary input values (6 and 2) and neuron outputs for two rows of neurons, while "updated" values show inputs and neuron outputs after dynamical reconfiguration step. We developed the Matlab DLLs to implement the I/O functions including read and write. We can see the results from the real experiment corresponding to VHDL simulation results as shown in Fig. 5. In this plot, "Enable_bus" representing the neuron selection signal, selects a particular neuron to be configured. Once the configuration process for all neurons is over, the outputs from neurons are stable and ready to be read out. This way we can update any neuron's configuration information without affecting the other neurons. In this example, we only update neurons 3 and 4 connections represented by "Enable_bus" content 4 and 5. The simulation results after updating correspond to the real experiments read back from hardware as the "updated" values in Fig.4.

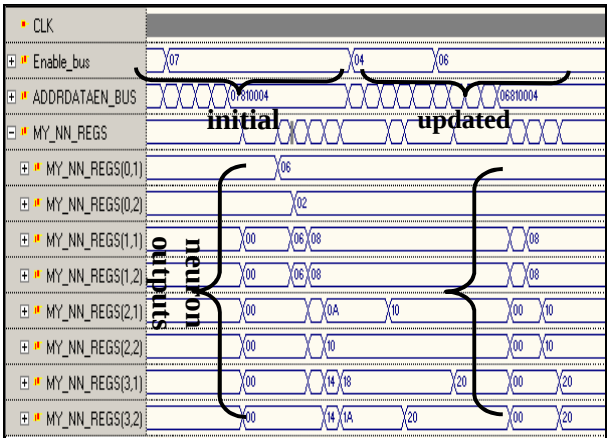

Fig. 5 VHDL simulation for partial configuration

The design has been described in VHDL and synthesized using the Xilinx XST and the Xilinx Alliance tools for place and route. The implementation results are summed up in Fig. 6.

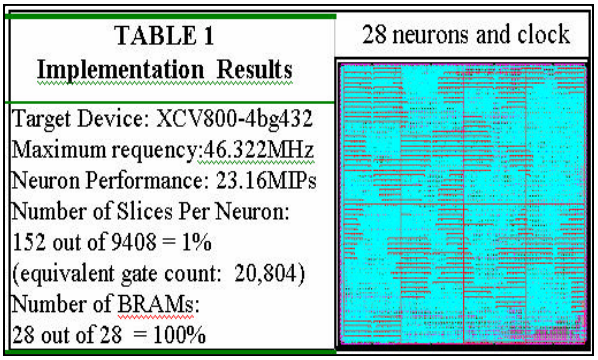

Fig. 6 Implementation report

The implementation results show that this neural network architecture realizes a maximum parallel instruction throughput of 23.16x28 MIPs with 28 fully connected neurons. The neuron number is limited to 28 since there are only 28 BRAM modules

on the chip although we still have some other resources unused. If we further lower the neuron's program memory size, we can put more neurons on a single chip to overcome the BRAM bottleneck.

The mapping result is shown in Fig. 7. We can see how the neurons are distributed inside the chip after the mapping process. Every single neuron occupies a compact and concentrated logic area. The compactness depends on the structure-oriented hardware design, for instance, we adopt LUTs and dedicated multiplexer module in addition to the optimized Picoblaze structure to build every single neuron.

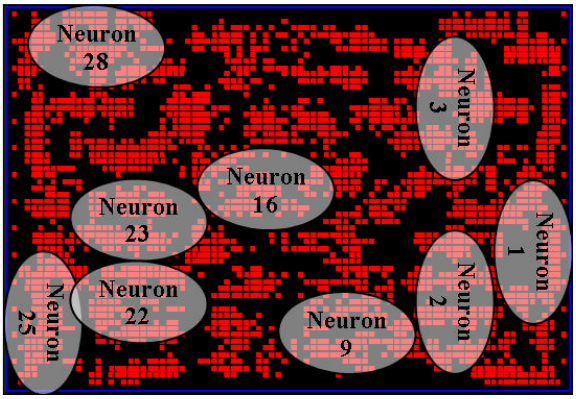

Fig. 7 Mapping result

5. Conclusions

This paper has described the architecture and chip implementation of an array of neurons aimed at implementation of SOLAR. The self-organizing learning is based on a new machine-learning algorithm [7] that combines knowledge from neural networks (NN) and information theory. SOLAR represents a new idea in hardware design of neural networks. It is modular and expandable system. It also defines a new breed of dynamically reconfigurable architectures that can dynamically reconfigure themselves. This presented architecture is a novel dynamically reconfigurable (via dual-port memory) neural network implementation based on simple general-purpose processor (KCPSM) architecture. Firstly, it has a regular expandable parallel architecture. Therefore, its speed and learning abilities can be greatly improved comparing to software simulation. Secondly, it has data-driven self-organizing learning structure based on the proposed self-organizing learning algorithm. Furthermore, design flexibility is attained by exploiting the features of self-reconfigurable neuron units. Finally, hardware reconfigurability is achieved in this self-organizing learning array by involving reconfigurable routing modules. According to the implementation results, this neuron architecture realizes a maximum parallel instruction throughput of 23.16x28 MIPs with 28 fully connected neurons. Much higher performance can be achieved by connecting more neurons. Hence it can be of practical use for embedded hardware applications in signal processing, wireless communications, multimedia systems, data networks, and so forth.

References

[1] Misra, M., 1997, Parallel Environment for Implementing Neural Networks. Neural Computing Survey, Vol.1, 48-60,1997.

[2] Glesner, M. and Pochmuller, W., 1994, An Overview of Neural Networks in VLSI, Chapman & Hall, London, 1994.

[3] G. Tempesti, D. Mange, A. Stauffer, C. Teuscher "The BioWall: an Electronic Tissue for Prototyping Bio-Inspired Systems", Proceedings, NASA/DoD Conf. on Evolvable Hardware, Los Alamitos, Calif., pp.221-230, 2002.

[4] L. Sekanina, "Towards Evolvable IP Cores for FPGAs," In: Proc. NASA/DoD Conf. on Evolvable Hardware, Los Alamitos, US, ICSP, 2003, pp. 145-154, 2003.

[5] Hugo de Garis, Michael Korkin, "The CAM-Brain Machine for Real Time Robot Control", J. Neurocomputing, Elsevier, Vol. 42, Issue 1-4, Feb., 2002.

[6] T.Roska et. al., "The Computational Infrastructure of Analogic CNN Computing – Part I.: the CNN-UM Prototyping System", *IEEE Trans. Circuits and Systems I*, vol. 46, pp. 261-268, 1999.

[7] J. A. Starzyk and T-H.Liu, "Design of a self-organizing learning array system," IEEE Int. Symposium on Circuits and Systems (ISCAS), Bangkok, Thailand, May 2003.

[8] J. A. Starzyk, and Y. Guo, "Dynamically Self-Reconfigurable Machine Learning Structure for FPGA Implementation" Proc. Int. Conf. on Engineering of Reconfigurable Systems and Algorithms (ERSA) Las Vegas, Nevada, USA, June,2003.

[9] K. Chapman,"8-Bit Microcontroller for Virtex Devices." Xilinx XAPP213, Online http://www.xilinx.com/xapp, Oct. 2000.

[10] Nallatech Ltd, "Ballynuey 2 VIRTEX PCI card user guide", 1993-1999.